

Final project: the U-Perseverance

Information:

- **Type:** team activity
- **Deadlines:**
 - **Follow-up:** weekly and on-demand
 - **Final delivery (demo day):** 6/13 of May (face 2 face, to be confirmed)
- **Max grade:** 10
- **Weight:** this activity represents a **40% of your final grade**.
- **Continuous assessment:** a minimum grade of **5 out of 10** is required to apply the continuous assessment rules.

Final project statement

This year we have seen how the NASA (and its partners) have again landed the Perseverance in Mars. This mission brings a lot of challenges and opens new possibilities up to explore and know more about the space. Apart from the scientific and technological objectives that the launch, landing and operation of the rover implies for the engineering teams, it is important to remark the impact that the mission is having in social media. It is possible to follow all the mission through the next link¹ and the rover itself has social profiles. Perhaps, this is the first time a robot (with a scientific purpose) has so much interaction in social networks implying the relevance of such a mission at a worldwide scale. From a technical perspective, this type of mission requires the collaboration of many engineering disciplines: experts and organizations around the world. Physics, mechanics, space, telecommunications, software, etc. must be orchestrated to build a safety-critical cyber-physical system.

This mission can serve us as motivation to learn more about software and one of its domain applications: robotics and control systems. The use of robots as a domain to learn programming has been demonstrated to provide added-value to the learning process motivating people to work collaboratively and to interact with elements, not just software but hardware, transforming the programming experience from a virtual world to something tangible in the physical world. That is why, we are going to use a robot (a rover) to apply the elements of programming and to provide an implementation to different tasks. In order to perform these tasks professionally, we will follow a basic software development lifecycle (analysis, design, implementation and test) including phases for operation (the demo day) and retirement (leave a robot ready for the next year).

As a final remark and due to the restrictions of COVID-19, it is necessary to strictly follow the safety instructions established by the university and the authorities regarding the use of working spaces, keep personal distance, etc. that will have some impact in the learning experience but still will allow us to build a cyber-physical system.

¹ <https://www.nasa.gov/perseverance>

To do so, it is strongly recommended that one person of the group (that may change over time) takes responsibility of the robot. The team works collaboratively and remotely producing the software and the person in charge of the robot makes the deployment and physical test.

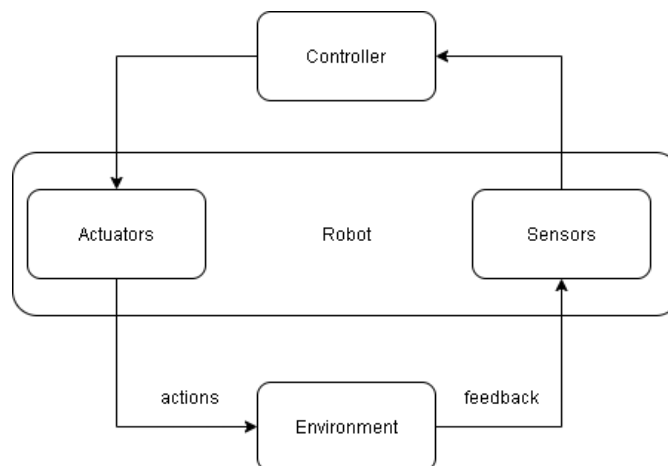
The GoPiGo rover

The GoPiGo rover is a robot specially designed for educational purposes with the possibility of using simple configurations (like Scratch and DexterOS) or more complex settings (like Python and Raspbian). The rover also includes many sensors although we only count with a distance sensor. In our case we will use the DexterOS to transfer the programs from a web and the rover will be powered by batteries (ensuring the safety use of the robot).

When programming robots, there is always a kind of control loop as it is presented below.

- Environment: space where the robot is running.
- Sensors: values measured returned from exploring the environment (e.g. a distance sensor).
- Controller: the software that is governing the robot.
- Actuators: mechanical elements that can be managed by the controller (e.g. a servo motor).

The different tasks will focus on programming a controller that takes as an input the values measured by sensors and perform some actions (e.g. find a hole). Other type of execution is configured when the rover waits for some user input to make a task (e.g. draw a figure).



For more detailed information, see the next official web <https://gopigo.io/>

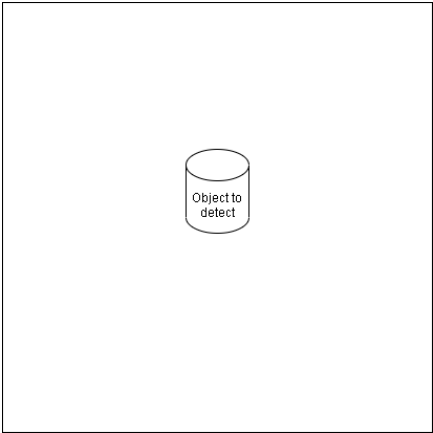
1. Task 1: Setting up and running

Id	1
Name	Setting up and running
Description	<p>The objective of this task is to assembly the rover and to be able to run simple programs. To do so, we will follow the steps established in:</p> <p>https://www.dexterindustries.com/GoPiGo/get-started-with-the-gopigo3-raspberry-pi-robot/1-assemble-gopigo3/</p>
Requirements	N/A
Implementation	N/A
Evaluation Criteria	The rover is up and running
Grade	0.5pt

2. Task 2: Drawing figures

Id	2
Name	Drawing figures
Description	<p>The robot shall be able to draw visible figures on the floor.</p> <p>A visible figure is such with a minimum size to make the rover to easily turn.</p>
Requirements	<ol style="list-style-type: none">1. The rover shall be able to draw a square of 4 sides with the same length.2. The rover shall be able to draw a circle.3. The rover shall be able to draw a rectangle.4. The rover shall be able to draw a triangle5. The rover shall be able to draw any geometrical figure taking as an input the number of sides.
Input	The type of figure and any other required parameter (e.g. size).
Expected output	The drawing of the figure on the floor.
Grade	0.5pt
Recommendations	<ul style="list-style-type: none">• Find out how to move the rover.• Try to create a simple figure.• Refactor the code to be able to draw any figure.• Make tests to ensure the proper behavior.

3. Task 3: Sentinel

Id	3
Name	Sentinel
Description	<p>The robot shall be able to explore some delimited space until finding a circular object.</p> <p>An object is found if it is less than X cm from the rover. In order to detect an object a scan (4 object detections) around the object would confirm the existence of an object and not just a wall. The object to detect could be situated anywhere in the space where the rover can maneuver.</p> 
Requirements	<ol style="list-style-type: none"> 1. The rover shall be able to explore a delimited space. 2. The rover shall be able to detect an object with a minimum of distance (between 10-20 cm) to allow the rover turn. 3. The rover shall confirm the existence of the object after making a scan around the object (min 4).
Input	The space to explore, the minimum distance and the number of scans to detect an object.
Expected output	<p>The detection of the object with a screen message or some log message.</p> <p>E.g. "The object has been found".</p>
Grade	2pt

4. Task 4: Save a plan of actions

Id	4
Name	Log and save a plan of actions
Description	<p>The robot shall be able to save the set of commands (movements) that have been executed. To do so, a domain specific language with 3 commands is proposed:</p> <pre> PROGRAM ::= SET TIME <PARAMETER> LIST_OF_COMMANDS LIST_OF_COMMANDS ::= COMMAND LIST_OF_COMMANDS COMMAND ::= MV <TYPE>, <PARAMETER>, <UNIT> STOP SET SPEED <PARAMETER> TYPE ::= FW BW R L PARAMETER ::= SIGN NUMBER NUMBER UNIT ::= C (centimeters) D (degrees) SIGN: + - </pre> <p>The time is set in milliseconds and corresponds to the waiting time between commands.</p> <p>Example:</p> <pre> SET TIME 2 //Mandatory SET SPEED 10 MV FW MV R, 90, D MV R, 90, D MV R, 90, D MV R, 90, D MV FW </pre>
Requirements	1. The rover shall be able to register its actions according to the 3 predefined commands,
Input	A filename (or the standard output).
Expected output	A set of commands serialized in a file, the standard output or other data structure (a list).
Grade	1pt

5. Task 5: Load and run a plan of actions

Id	5
Name	Load a plan of actions
Description	The robot shall be able to read until 10 plans identified by a name (filename) of commands and execute them.
Requirements	<ol style="list-style-type: none">1. The rover interpreter shall be able to read a file of commands.2. The rover interpreter shall validate all commands before starting the execution. If any error is found a message shall be displayed in the console cancelling the execution.3. The rover interpreter shall execute the actions command by command waiting a established time between commands (500-2000 ms).4. The rover interpreter shall validate that the command can be executed safely (without hitting any object). (Optional)
Input	<p>A filename or a list of commands identified by an unique identifier.</p> <p>EXAMPLE:</p> <pre>SET TIME 500 SET SPEED 10 MV FW MV R, 90, D MV R, 90, D MV R, 90, D MV R, 90, D MV FW</pre>
Expected output	The execution of the commands.
Grade	2pt

6. Task 6: Scape room

ID	6
Name	Scape room
Description	The robot shall be able to go through some circuit and find the exit gate.
Requirements	<ol style="list-style-type: none"> 1. The rover shall be able to follow a path (forward movements) and detecting the limits to avoid collisions and to be able to turn in some direction and continue. 2. The rover shall be able to start in some point and continue until the exit gate. <ol style="list-style-type: none"> a. The exit gate could be implemented as a rule of 10 forward movements without limit detection. (Optional)
Input	<p>A circuit (delimited space, side borders) with enough space to maneuver.</p>
Expected output	The rover exits the circuit without any collision.
Grade	<p>2pt</p> <p>1pt if the rover is able to go all the way.</p> <p>0-1 pt depending on the time. Quartiles: Q1-1pt (first two), Q2-0.75 (second two), Q3-0.5 (third two), Q4: 0.25 (last one)</p>

7. Task 7: Open task

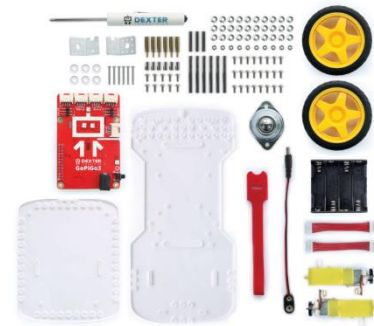
Id	7
Name	Open task
Description	<p>This task is open to innovative ideas. The combination of the different elements of programming and the capabilities of the rover will be evaluated.</p> <p>Potential ideas:</p> <ul style="list-style-type: none">• A menu to manage the rover with commands• Extensions in the language to manage the rover• New capabilities exploring a space• Do (forward) and un-do (backward) the same path• An interpreter to send commands to the rover• ...
Requirements	
Input	
Expected output	
Grade	1pt

Following the same dissemination strategy as the Perseverance, I would like to encourage you to post in social media the evolution of your rover. For instance, using Twitter posts with images or short videos with the next hashtags: #programming #roveruc3m #uc3m

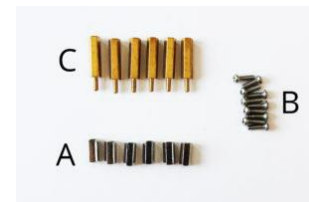
8. Bill of Materials

The GoPiGo rovers have been acquired for this course. It is a kind of course heritage so, everybody must take care of the materials to be reused in other academic years. Issues, failing parts, etc. can appear due to everyday use but we must try to minimize them using all materials carefully.

- GoPiGo3 Base Kit
 - 2 Yellow Wheels
 - 2 Motors
 - 2 Motor Cables
 - GoPiGo3 Board (red)
 - 2 Metal Motor Brackets
 - 1 Metal Caster Wheel
 - 2 Large Acrylic Pieces (body and canopy)
 - 1 Battery Box
 - 1 Battery Cable
 - 1 Velcro Strap
 - Small hardware bag:



- 1 Small Screwdriver
- (A) 6 Short Silver Posts
- (B) 10 Mini Screws



- Hardware bag:
 - (D) 24 Short Screws
 - (E) 30 Washers
 - (F) 10 Nuts
 - (G) 6 Long Screws
 - (H) 6 Round Acrylic Spacers
 - (I) 3 Medium Silver Posts
 - (J) 6 Long Silver Posts



- Raspberry Pi (green computer board)
- SD Card with Dexter Industries software
- USB Drive (if using DexterOS SD card)
- 8 AA Rechargeable Batteries and a hub for recharging batteries.
- An America-Europe adapter.

9. Relevant links

- <https://gopigo3.readthedocs.io/en/master/>
- <https://www.dexterindustries.com/gopigo3-tutorials-documentation/>
- <https://readthedocs.org/projects/gopigo3/downloads/pdf/latest/>
- <https://github.com/DexterInd/GoPiGo>
- <https://github.com/DexterInd/GoPiGo/tree/master/Software/Python>

10. Summary of evaluation criteria

Task	Grade
Task 1: Setting up and running	0.5
Task 2: Drawing figures	0.5
Task 3: Sentinel	2
Task 4: Log and save a plan of actions	1
Task 5: Load and run a plan of actions	2
Task 6: Scape room	2
Task 7: Open task	1
Quality of the implementations (clarity of code, application of the programming techniques, test cases, etc.)	1
Total	10

10.1. Deliverables and submission

The team leader shall submit a zip file, named as **<COLOR>.zip**, with the following contents in the corresponding task:

- The Python source code (including comments and documentation):
<COLOR>/src/task-X/*.py
- Any other file used in the development or videos as evidence.